项目名称 Project Name		密级 Confidentiality Level
		仅供收件方查阅
项目编号 Project ID	版本 Version	文档编号 Document Code
1	1.0	1_SD_001

APP
架构设计说明文档

拟制：		日期：	2020.7.7
评审：		日期：	
批准：		日期：	

Revision Record

修订记录

日期	修订版本	CR/ Defect 号	修改章节	修改描述	作者
2020-9-27	初稿				

目录

一、简介

1.1、目的

该文档的读者主要为项目的开发人员、项目经理以及文档审核人员。

设计该文档的目的在于以下的几个方面：

1、 **有利于信息持有者之间的沟通：**体系结构是一个系统的个高层表示，可以作为不同信息持有者之间信息讨论的焦点，有利于开发人员对整个系统有更好的把握，更好的进行开发和交流。

2、 **便于系统分析：**在项目开发的早期阶段给出系统的确切结构，实际上就是对系统的分析过程。体系设计决策对系统额能否满足关键性需求（如系统的功能、可靠性、可维持行等）有着重要的影响。

3、 **有利于后续项目开发：**系统设计为后续的项目开发提供了模板，既有利于开发人员对系统整体的把握，又为开发人员确定规范了开发结构，开发人员在这个体系上开发项目，既有利于后续的管理，又有利于最后的整合。

1.2、范围

1.2.1、名称

1.2.2、应用

主要的应用领域为非专业音乐领域，采用线上 app 的方式。

该系统旨在为用户提供一个自由便捷的音乐制作软件。在该平台上，即便是不具备音乐基础的人，也可以在该软件简单的引导下完成一首简单歌曲的自主创作。

1.3.1、背景

在现在的社会上，音乐的传播范围越来越广，热爱音乐的人越来越多。但大部分人仅局

限于听音乐，制作音乐还是属于那些经过一定的专业教育、有一定音乐素养的人群，普通的群众也仅限于翻唱他们写的歌曲。

通过这款 APP，一个人仅仅通过哼唱也可以制作一处交响乐的效果。并且在如今，唱作已经不再是专业音乐人的特权，年轻人对音乐已随着近些年娱乐产业与精神文化的提升愈发渴求，通过如此简单的方式实现音乐制作，无疑是给了无数年轻人圆了一个音乐制作人的梦想。

二、需求分析

2.1、功能性需求分析

可以将多段音频进行合成一整段音频，每段音频的导入来源有“用户哼唱”、“本地导入”、“简谱识别”。在加入每段音频之后，可以进行微调，进行裁剪、音量调节、音高调节。

在音频合成完毕之后，可以对自选音频段进行伴奏添加。可选的伴奏乐器有“钢琴”“吉他”“架子鼓”“贝斯“。

（2）互动分享

同样，对于只是想听歌的朋友，我们也会制作一套推荐系统来根据其平时的听歌喜好（如摇滚，soul，电子或民谣等）对其进行智能歌单的推送。并且用户也可以对喜欢的作品进行点赞打赏，并且分享动态（可以形成像微博一样的社交环境，并且可以引入如同知乎大 V 一样的专业人士对专业知识进行点评与回答）

2.2、质量属性

可用性：

1. 系统达到较高可用性标准，每年因为系统故障以及系统更新而停用的时间小于等于 8.8 小时，达到可用性级别 99.9%。
2. 在正常操作下，系统数据库出现错误时，在一小时内完成自动更换备份数据库。
3. 在正常操作下，系统出现错误或异常但不影响操作结果时，系统给出警告信息并保持运行。

性能：

1. 通过“用户哼唱”方式导入音频时，从用户开始上传到系统接受用户音频的时间间隔不超过 0.4s。
2. 通过“本地上传”方式导入音频时，从用户开始上传到系统接受音频文件的时间间隔不超过 0.7s。
3. 在用户导入简谱的时候，从用户上传到系统接受的时间间隔不超过 1s。
4. 系统将用户上传的音频解码，转换成系统可以使用格式的时长不超过 4s。
5. 系统将简谱识别，然后转换成系统可以使用格式的时长不超过 5s。
6. 在用户对小段音频进行微调时，系统处理用户操作的时长不超过 0.3s。
7. 在用户添加钢琴、架子鼓伴奏时，(系统添加伴奏的处理时间:伴奏时长)<(1:30)
8. 在用户添加吉他伴奏时，(系统添加伴奏的处理时间:伴奏时长)<(1:20)
9. 用户评论、转发、点赞的操作，应该在 0.3s 内处理完成，并体现在界面上。
10. 在用户编辑完毕之后，生成最终音频文件的时长不超过 30s。

安全性：

1. 提供冗余的系统数据存储和数据备份，对数据库进行在线数据备份，当数据库遭到破坏用户数据丢失时，可在 3 分钟内恢复全部数据和系统的正常使用。
2. 用户进行在线支付时，系统可以阻止 100%的病毒攻击，防止用户交易信息泄露和资金被非法转移。
3. 系统使用限流阀机制和过期机制，以及辅助检测手段防止 Dos 攻击。
4. 屏蔽某 IP 短时间内大量无意义访问，以防正常用户无法使用。
5. 用户进行提现操作时，提现的金额应该安全转移到用户的收款账户中，在退款过程中拦截 100%病毒攻击防止资金非法转移。

易用性：

1. 当用户进行错误操作后，可在运行时 1 秒内取消错误操作使影响最低。
2. 所有具有基本手机操作常识的人，可在 3 分钟内根据界面设计掌握本系统的基本使用方法，熟练用户可以使用快捷键进行操作。
3. 在用户编辑音乐时，对于不同的伴奏类型应该采用不同的图标进行标注，而且每种图标的特点可以鲜明的反应伴奏的类型，除此之外，在图标下应该提供一定的文字提醒。

4. 在用户使用音频编辑合成功能时，应该为用户提供一个“新手指引”选项，在第一次使用时会弹出提醒，在第二次使用时，该“新手指引”选项会放在界面右上角，不在弹出提醒。
5. 用户操作易于记忆，用户在使用本 app 三次之后，可以熟悉记起其基本操作步骤，不需要重新学习。
6. 系统错误率为，新用户正常进行系统操作时，平均出错率低于 15%，熟练用户正常进行系统操作时，平均出错率低于 2%。
7. 用户进行账号注册和账号登录等输入操作时，若出现输入错误或输入内容不当情况，系统利用界面文字提醒用户及时更正，使用户每次出错平均更正时间低于 2 秒。
8. 当用户需要查看帮助或进行交互时，可读的排版和易用的导航可以为用户提供其所需要的信息。例如在帮助界面等将重要信息标红。
9. 对于全部用户操作系统在 2 秒内提供文字或声音反馈，并实现合适的界面跳转。

可修改性：

1. 用户希望调整系统界面，在维护时，开发人员只修改界面及相关配置文件，不影响其他部分，能在 5 小时内完成修改并发布更新。
2. 保证系统的开放性，只要符合系统规范，可以简单的加入或减少系统模块，可以通过软件的修补完成系统升级和更新。

可移植性：

1. 在新的平台上运行的需求出现，开发人员需要修改代码，使之与新环境适配，只需要修改与目标环境相关的适配层代码，不用修改与目标环境无关的主体层代码，最小程度上降低了修改量，可以在一周内完成。

可测试性：

1. 开发人员在设计时统一做一个操作面板，其为一个可操作性整个系统的独立模块，并提供两种形式，一种为命令形式执行操作并获得基本输出，另一种为 GUI 形式，在界面上展现输入输出。
2. 单元测试人员在组件完成时执行单元测试，系统内的各组件有控制行为的接口并且其输出可观察，例如测试用户提交音频文件之后是否可以及时的转译为系统可以使用的文件类型等。可以在 3 小时内测试 85%的路径。

3. 系统用户在使用所交付的系统时，对完整的应用进行测试。系统可以对出现的错误或故障进行详细提示，使测试结果易于判断，有可分析性、可获得性。且出现故障的概率小于 0.1%。
4. 提供事务日志使过去的系统状态和变量可见，在运行中可查询。

可维护性：

1. 设计人员在设计时尽可能地将不同模块分离，降低后合度，从而使后面维护人员更容易地对单独的模块进行修改，替换和升级。
2. 设计人员在设计时撰写详细的设计和帮助文档，以便后续的维护人员更好的理解系统并进行维护。

2.3、限制条件

技术限制：

① 该软件必须运行在 Android 系统上。

② 需要实现系统框架，对使用语言形式不限，可以是 C、java 等；

③ 开发成员对 java 更为熟悉，所以使用 java 进行开发；

④ 使用 InteliJ IDEA 进行系统程序编写；

⑤ 使用 MySQL 数据库进行数据的存储和读取；

法律限制：

① 所有软件都选用正版；

② 所有技术资料由开发一方保管；

③ 合同制定确定违约责任；

社会限制：

① 要维护其他音乐人的版权

商业限制：

① 完成时间要求：在一年之内完成该系统

② 成本、预算：需要花费小组成员的大量时间精力，在合理调配小组成员的精力和时间的基础上实现系统的完成和完善；

③ 预计的系统生命周期的长短：预计系统周期为 4 年，最长可达 5 年；

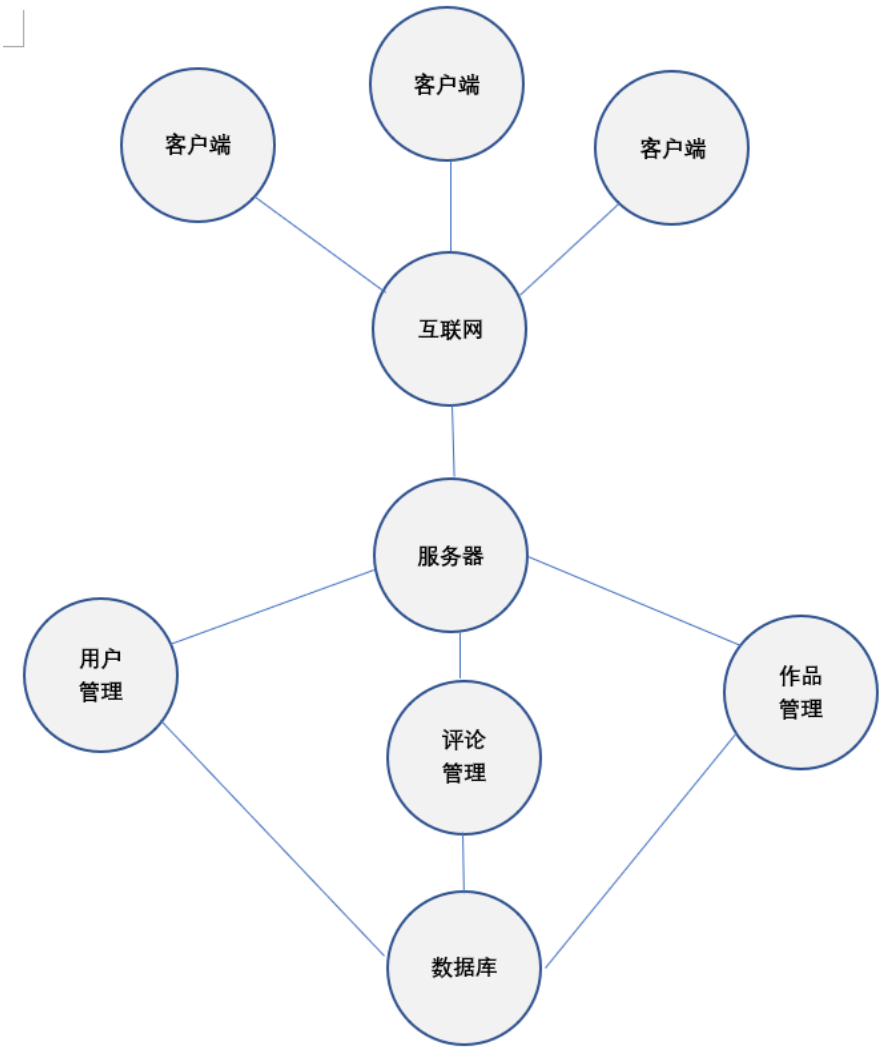
④ 目标市场：面向所有人，所有想要使用该软件的人，在同意使用条约的情况下，均可以使用。

⑤ 推出计划：首先实现一个基础的系统，在此基础上不断完善优化，或是增加新的功能；

⑥ 集成：不与老系统集成，实现一个全新的系统。

⑦ 该 app 要保证用户的版权，当用户的音频作品非法用于商业用途时，要能够维护用户的合法权益。

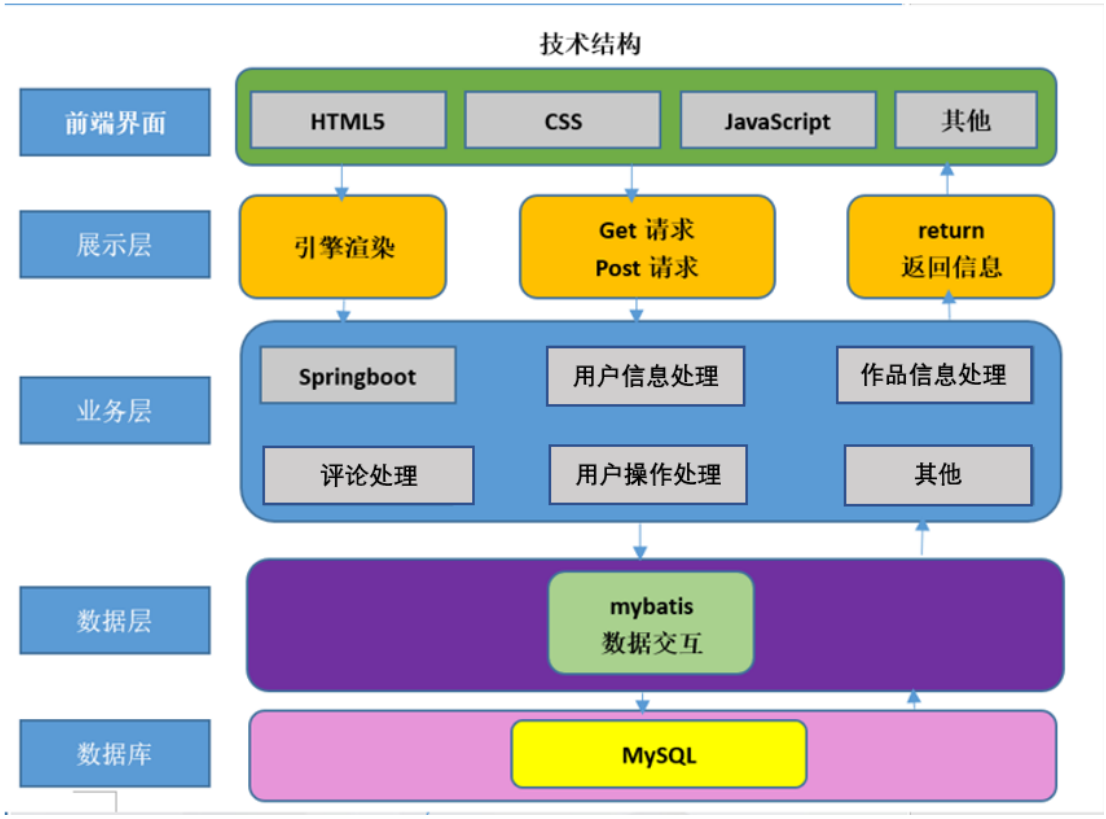
三、系统场景



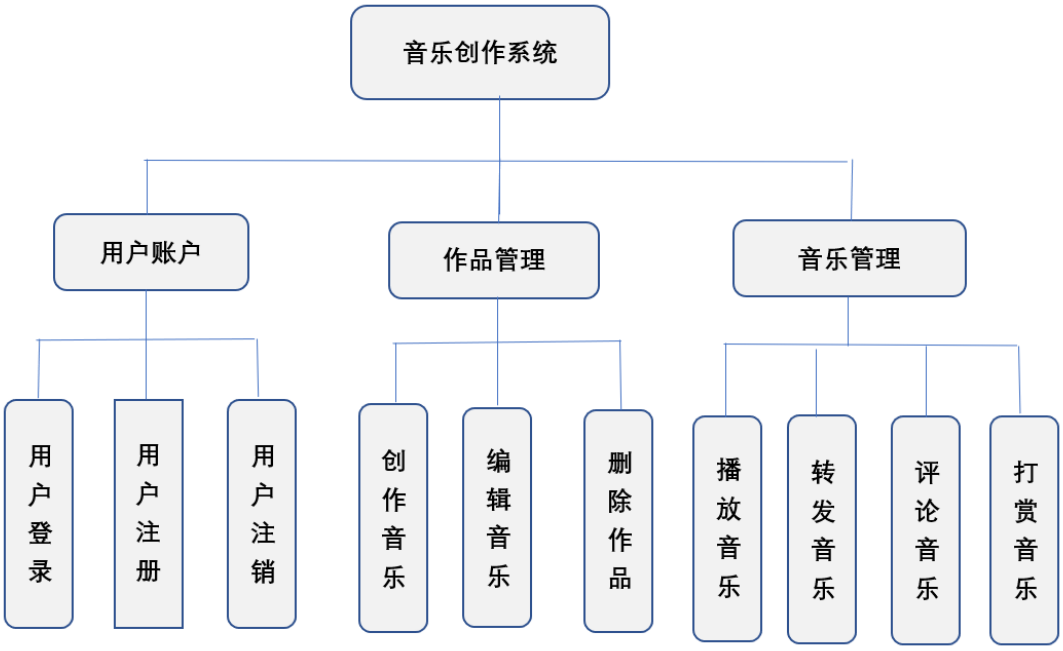
四、软件架构设计

4.1、系统分层架构设计

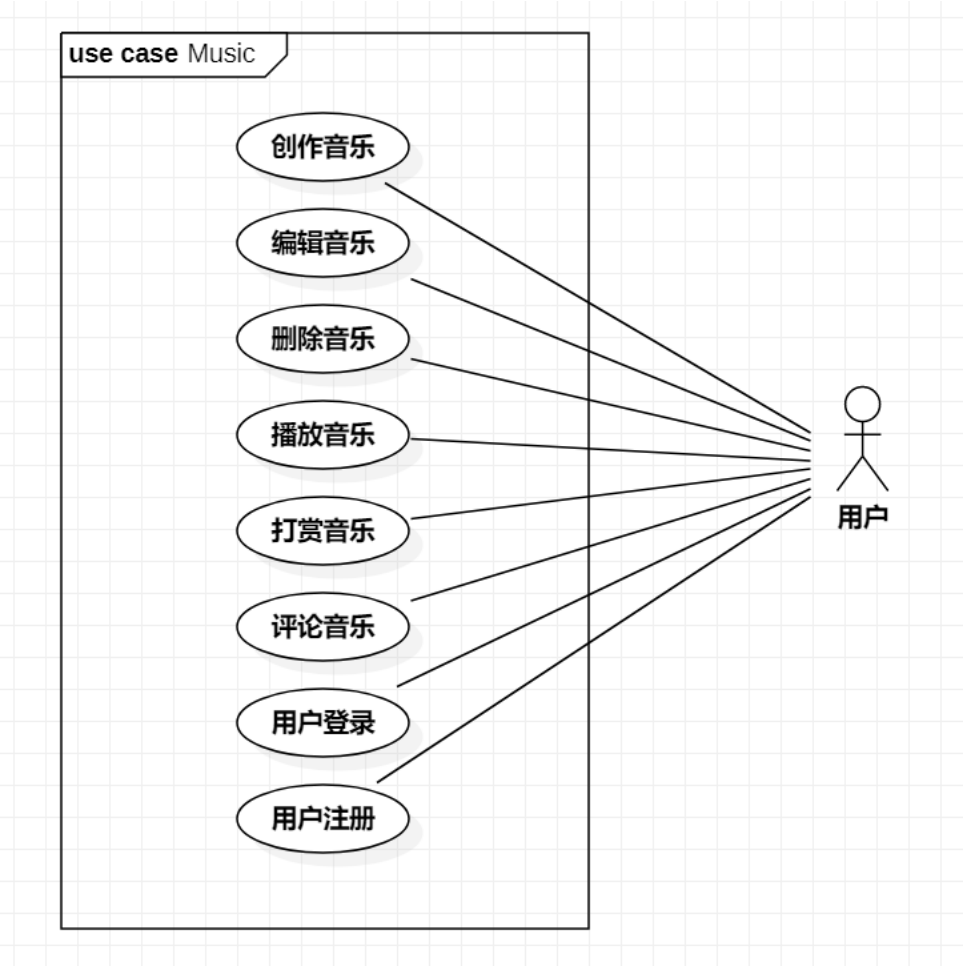
4.1.1、技术结构



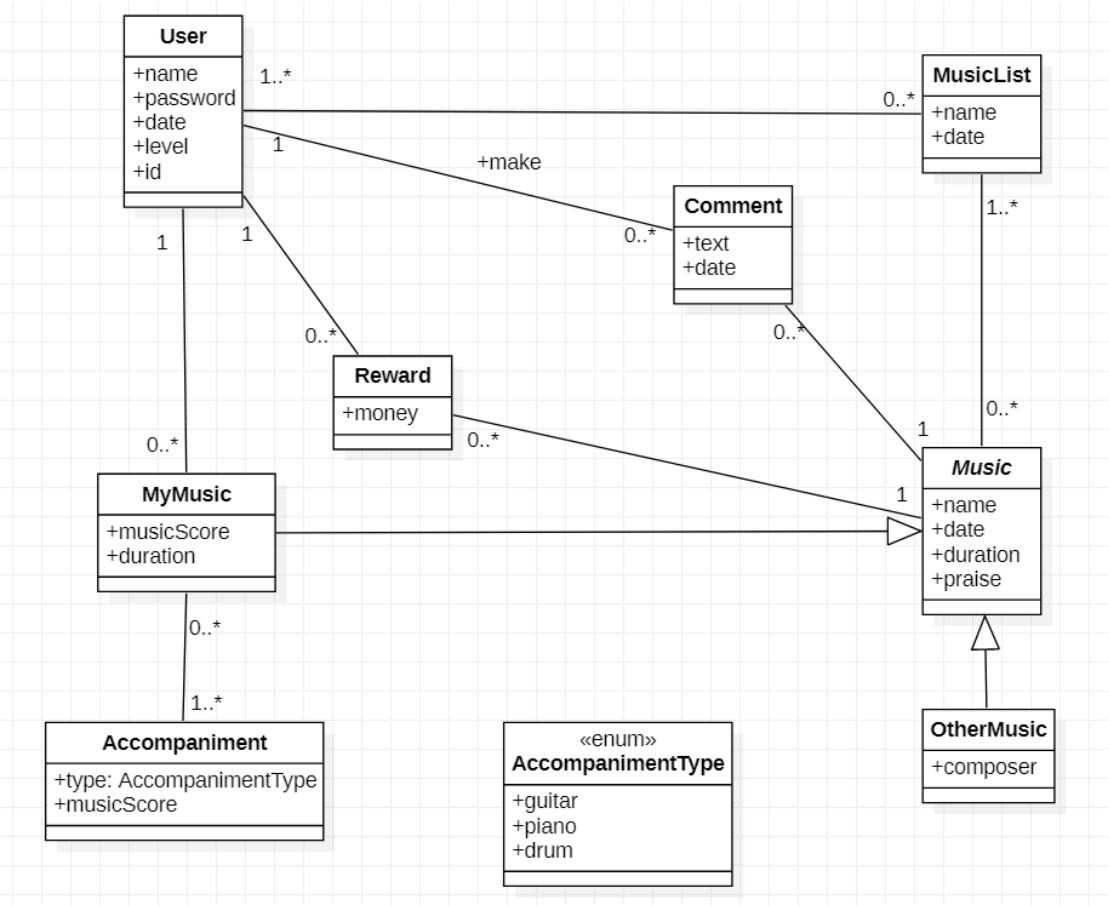
4.1.2、功能结构


音乐创作系统
用户账户
作品管理
音乐管理
用户登录
用户注册
用户注销
创作音乐
编辑音乐
删除作品
播放音乐
转发音乐
评论音乐
打赏音乐

4.2、用例视图

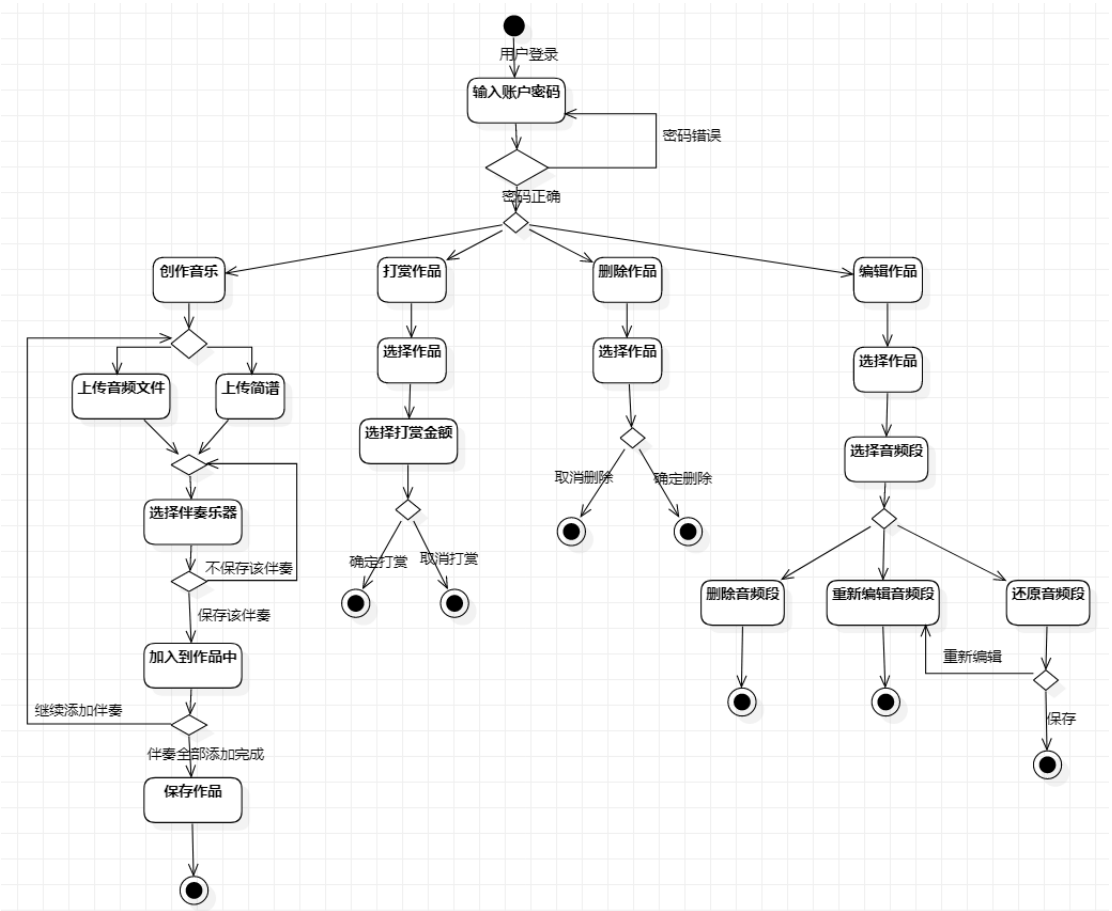


4.3、逻辑视图

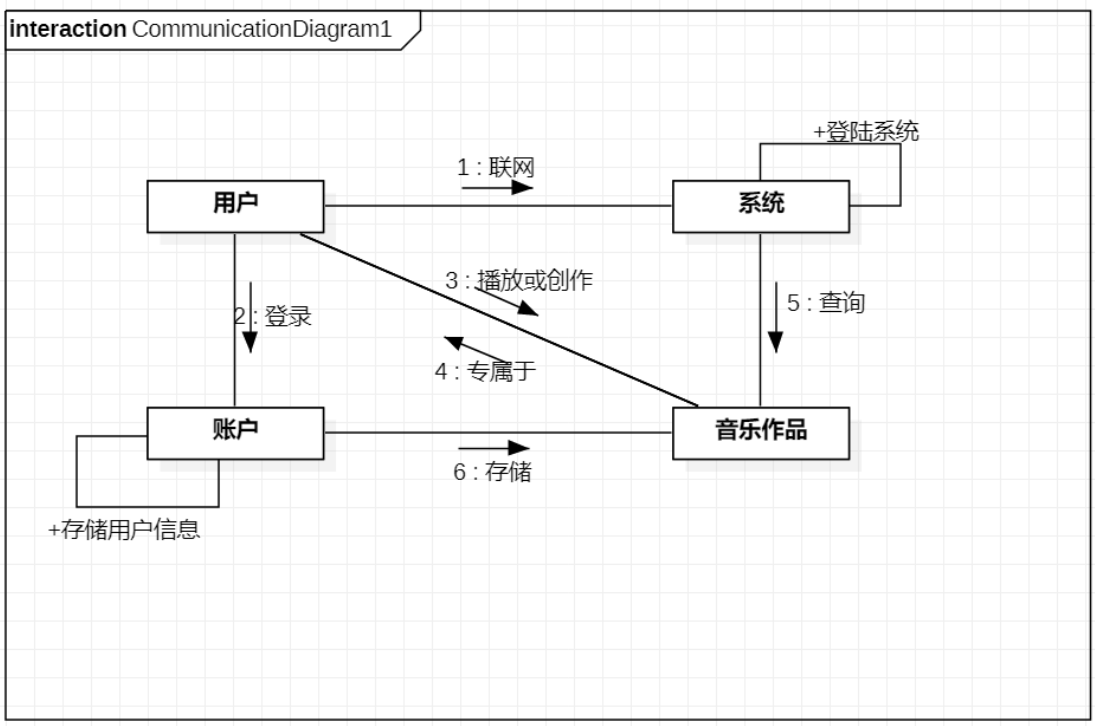


4.4、过程视图

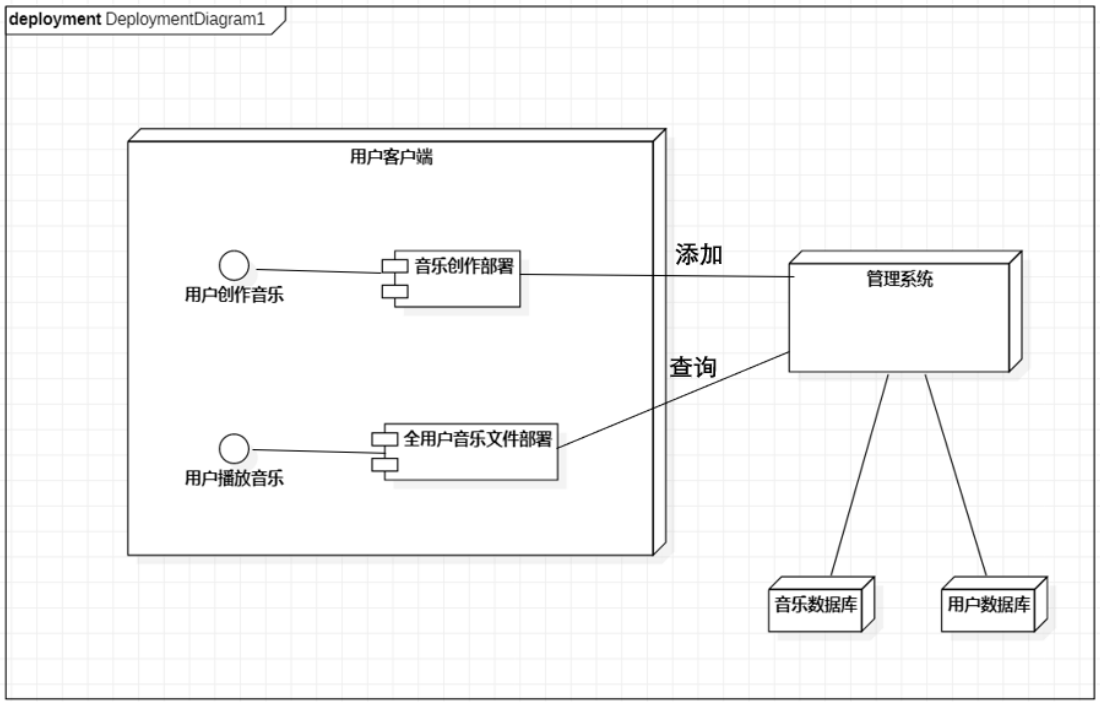
用户操作活动图：



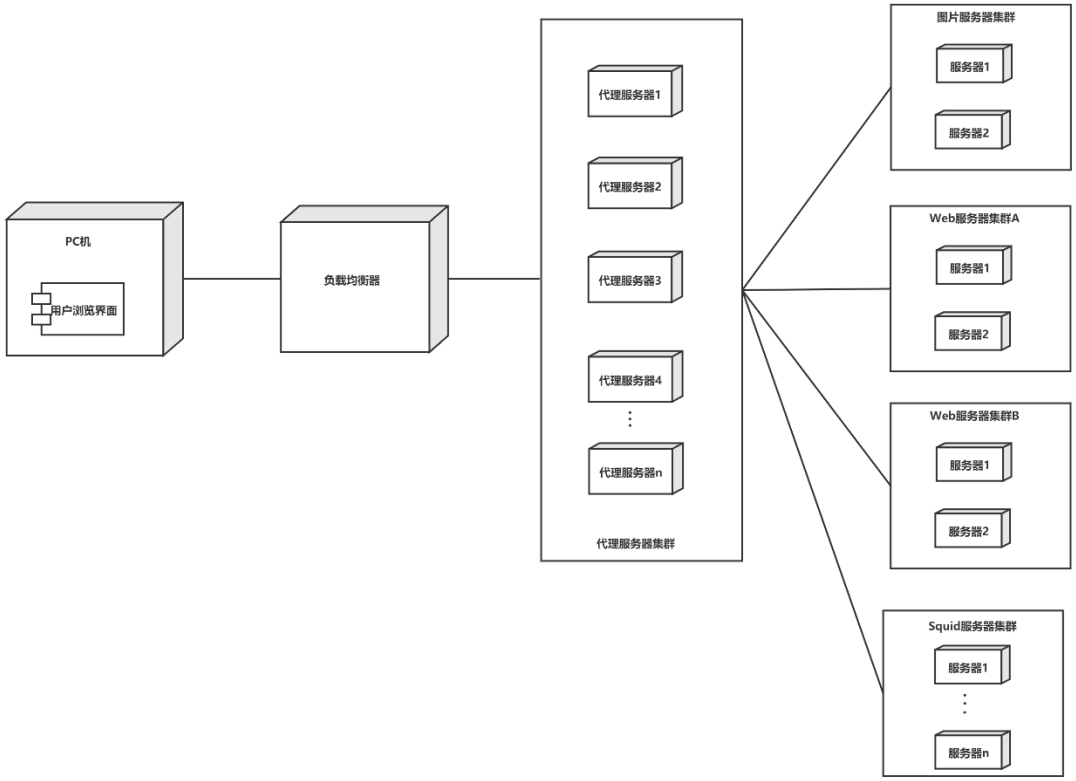
Communication Diagram：



4.5、部署视图



4.6、物理视图



五、关键技术设计

5.1、接口设计

5.1.1、前后端用户信息对接

1. 在前端获取用户信息之后，通过接口传递给后端。
2. 给后端传递一个用户类 User，里面包含了用户的基本信息。
3. 异常定义：

传递的信息为空。

网络故障时，信息传输失败。

前端在获取用户信息时，出现参数的获取错误。

4. 质量属性特征：

在 10000000 次对接中，最多容忍出错的次数为 1 次。

出错之后，能够在 10 分钟内测试出错误，在 30 分钟内做出修正。

从信息发出，到后端接收到接口的消息，这个时间间隔不应超过 1ms。

能容许 10 万人同时进行接口使用。

信息的传送方式采用加密通信，提升安全性，

在出现错误时，要能及时给用户发出提醒。

5.1.2、前后端音频信息对接

1. 系统在获取用户选择的音乐文件时，通过接口将该音乐的识别信息传递给后端。
2. 后端反馈给前端一个 Music 类，里面包含了该音乐的基本信息。
3. 异常定义：

传递的信息为空。

在音乐识别信息传输过程中，出现信息丢失。

网络故障时，信息传输失败。

前端在获取音乐识别信息时，出现参数的获取错误。

5. 质量属性特征：

在 10000000 次对接中，最多容忍出错的次数为 1 次。

从信息发出，到后端接收到接口的消息，这个时间间隔不应超过 1ms。

接口的最大并发数为 100 万。

无法查询到音乐信息时，要给用户及时提醒。

出错之后，能够在 10 分钟内测试出错误，在 30 分钟内做出修正。

5.1.3、软件与麦克风对接

1. 用户使用麦克风录取音频。
2. 系统接受请求之后，向硬件发出请求，在获得接口使用权限之后，获取接下来录入的一段音频，并将音频文件传输到软件系统当中。
3. 异常定义：

无法获得麦克风权限。

麦克风获得的音频文件无法传输到 app 内。

从麦克风获得的音频文件在传输过程中出现错误。

4. 质量属性特征：

在用户申请使用麦克风时，需要在 1s 之内完成响应。

将音频文件传输到 app 内所消耗的时间不超过 3s。

音频传输内容的正确率 99.99%。

在使用接口时，要对用户进行适当的提示，提示用户何时完成音频文件的传输。

在接口出现问题时，要给用户弹出提示。

5.2、数据结构

1. 用户类：User

   功能：存储用户的具体信息

   具体变量：

   id：存储用户唯一的编号 id

   name：存储用户的昵称

   password：存储用户的密码

   date：存储用户创建的日期。

level：存储用户的等级

MyMusicList：存储用户自主创作歌曲集合

MyFavouriteMusic：存储用户喜欢的歌曲

2. 用户创作音乐类：MyMusic

   功能：存储用户创作作品的具体信息

   具体变量：

   name：作品名称

   date：创作日期

   praise：获赞数

   reward：打赏数

   composer：创作人

   duration：作品时长

   musicScore：乐谱

   accompany：存储每个音频段对应的伴奏

3. 伴奏类：Accompanyment

   功能：存储没一段音频文件对应的伴奏

   具体变量：

   type：伴奏类型

   musicScore：乐谱

4. 非本用户音乐：OtherMusic

   功能：存储此用户看到的其他用户音乐的信息。

   具体变量：

   name：作品名称

   date：创作日期

   praise：获赞数

   composer：创作人

   duration：作品时长

5. 用户评论类：Comment

   功能：存储用户对音乐的评论

   具体变量：

maker：评论音乐的人

text：评论的正文

music：评论的音乐

5.3、数据库设计

user:（用户数据库）

字段	类型	可为空	描述
id	Vchar	NO	用户 id，自增，为主键
name	Vchar	NO	用户名
password	Vhcar	YES	用户密码
level	Vchar	YES	用户等级
date	Vchar	YES	用户创建时间

collect_music_list_tb:（收藏歌单数据库）

字段	类型	可为空	描述
collectSongListId	Integer	NO	收藏在收藏歌单中的 id
userId	Integer	NO	收藏该歌单的用户 id
songListId	Integer	NO	歌曲在所有歌单中的 id

music_list_comment_tb:（歌单评论数据库）

字段	类型	可为空	描述
songListCommentId	Integer	NO	评论的 id 号
userName	Integer	NO	评论的用户名
songListId	Integer	NO	被评论的歌单 id
comment	Varchar	NO	评论的内容
createTime	Varchar	NO	评论发表的时间

music_list_song_tb:（被存放进歌单的歌曲数据库）

字段	类型	可为空	描述
songListSongId	Integer	NO	被收藏歌曲在该数据库中的 id
songId	Integer	NO	歌曲本身的 id
songListId	Integer	NO	歌曲被存进的歌单 id

music:（歌曲数据库）

字段	类型	可为空	描述
id	Integer	NO	歌曲的 id，自增
songName	Integer	NO	歌曲名称
composerID	Integer	NO	创作者 ID
date	Varchar	NO	创作日期
Lyric	text	NO	歌词
songComment	varchar	YES	歌曲评论
praise	integer	NO	作品获赞量

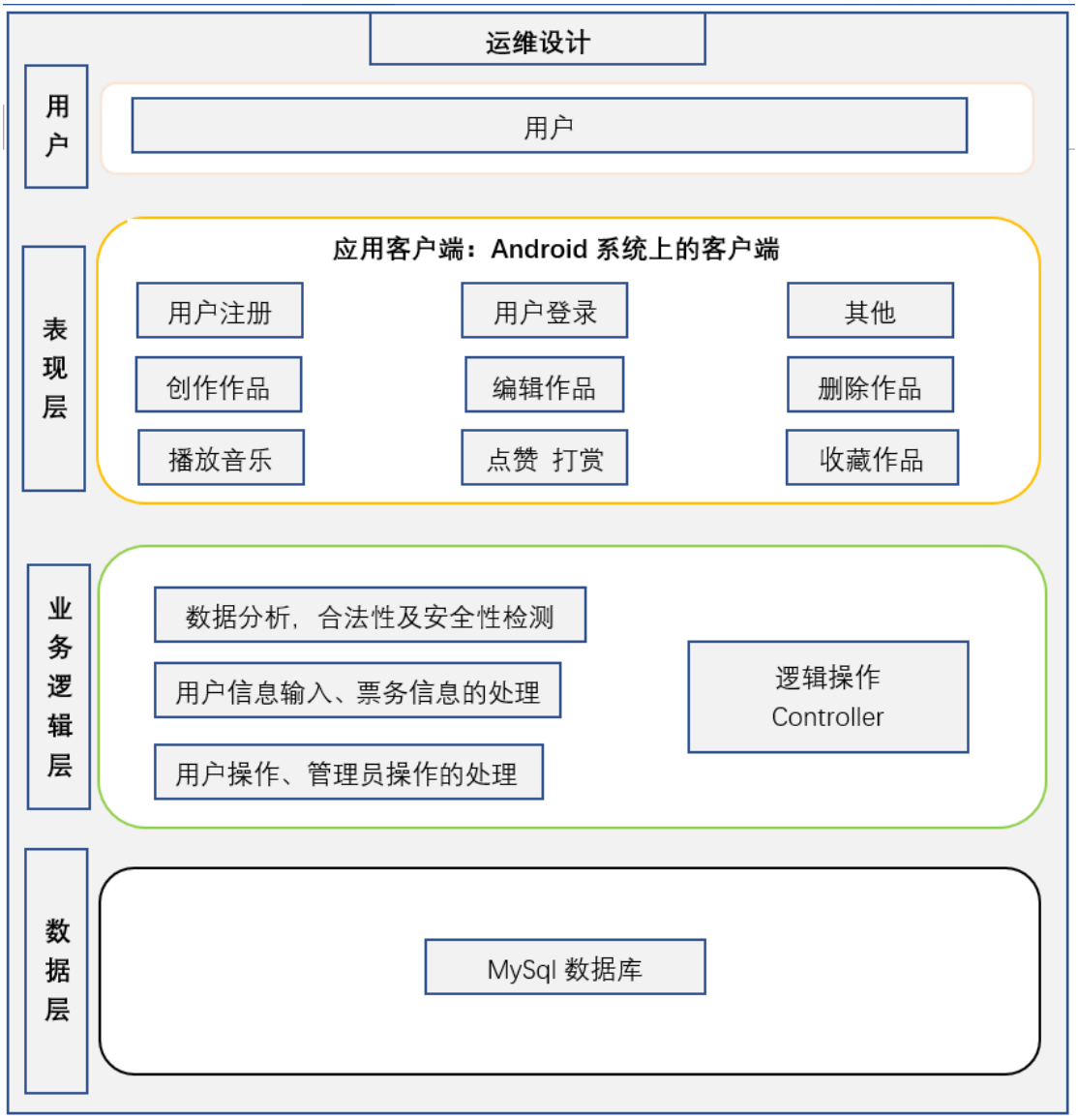
musicComment:（歌单评论数据库）

字段	类型	可为空	描述
id	Integer	NO	评论的 id 号
songComment	varchar	NO	评论的具体信息
userId	Integer	NO	发表评论的用户
createDate	varchar	NO	评论发表的时间

accompanyment:（歌单评论数据库）

字段	类型	可为空	描述
id	Integer	NO	该段伴奏的 id
musicID	Integer	NO	伴奏对应的音乐
address	vachar	NO	该伴奏文件的存储地址
createDate	varchar	NO	评论发表的时间

5.4、运维设计



六、其他

6.1、优先级

功能名称	优先级	功能名称	优先级
用户注册	A	点赞打赏	B
用户登录	A	评论作品	B
创作作品	A	播放音乐	A
编辑作品	B	收藏音乐	B

删除作品	B	转发作品	C

（注：优先级顺序为 A>B>C）